



## AMENDMENTS TO THE SPECIFICATION

Please delete the paragraph beginning with "The profiler agent can . . ." at the bottom of page 20 of Appendix A of the specification and ending with ". . . GetEnv call" at the top of page 21 of Appendix A.

Please add the following paragraphs after the heading "1.2. Function Call Interface" on page 20 of Appendix A of the specification:

The profiler agent can obtain a function call interface by issuing GetEnv call on the JavaVM pointer. For example, the following code retrieves the version of JVMPI interface that is implemented in JDK 1.2:

```
JVMPI_Interface *jvmapi_interface;

JNIEXPORT jint JNICALL JVM_OnLoad(JavaVM *jvm, char
    *options, void *reserved)
{
    int res = (*jvm)->GetEnv(jvm, (void
*)&jvmapi_interface,
    JVMPI_VERSION_1);
    if (res < 0) {
        return JNI_ERR;
    }
    . . .      /* use entries in jvmapi_interface */
}
```

The JVMPI\_Interface structure defines the function call interface between the profiler agent and the VM:

```
/* interface functions */
typedef struct {
    jint version; /* JVMPI version */

    /* -----interface implemented by the profiler-----
    */

    void (*NotifyEvent)(JVMPI_Event *event);

    /* -----interface implemented by the JVM----- */

    jint (*EnableEvent)(jint event_type, void *arg);
    jint (*DisableEvent)(jint event_type, void *arg);
    jint (*RequestEvent)(jint event_type, void *arg);
}
```

```
void (*GetCallTrace)(JVMPI_CallTrace *trace,
    jint depth);

void (*ProfilerExit)(jint);

JVMPI_Rawmonitor (*RawMonitorCreate)(char *lock_name);
void (*RawMonitorEnter)(JVMPI_RawMonitor lock_id);
void (*RawMonitorExit)(JVMPI_RawMonitor lock_id);
void (*RawMonitorWait)(JVMPI_RawMonitor lock_id,
    jlong ms);
void (*RawMonitorNotifyAll)(JVMPI_RawMonitor lock_id);
void (*RawMonitorDestroy)(JVMPI_RawMonitor lock_id);

jlong (*GetCurrentThreadCpuTime)(void);
void (*SuspendThread)(JNIEnv *env);
void (*ResumeThread)(JNIEnv *env);
jint (*GetThreadStatus)(JNIEnv *env);
jboolean (*ThreadHasRun)(JNIEnv *env);
jint (*CreateSystemThread)(char *name, jint priority,
    void (*f)(void *));
void (*SetThreadLocalStorage)(JNIEnv *env_id,
    void *ptr);
void * (*GetThreadLocalStorage)(JNIEnv *env_id);

void (*DisableGC)(void);
void (*EnableGC)(void);
void (*RunGC)(void);

jobjectID (*GetThreadObject)(JNIEnv *env);
jobjectID (*GetMethodClass)(jmethodID mid);
} JVMPI_Interface;
```

The `GetEnv` function returns a pointer to a `JVMPI_Interface` whose version field indicates a JVMPI version that is compatible to the version number argument passed in the `GetEnv` call. Note that the value of the version field is not necessarily identical to the version argument passed in the `GetEnv` call.